Mobile Robot Navigation Error Handling Using an Extended Kalman Filter

Aydin Saderzadeh
Mechatronics Research Lab
Faculty of Electronic, Computer Engineering& IT
Islamic Azad university, Qazvin Branch
Qazvin, Iran
E-mail:aydin_saderzadeh@yahoo.com

M.Mehdi Sanaatiyan
Mechatronics Research Lab
Faculty of Electronic, Computer Engineering& IT
Islamic Azad university, Qazvin Branch
Qazvin, Iran
E-mail:m.sanaatiyan@gmail.com

M.Habbibnejad Korayem
Faculty of Mechanic Engineering
Iran University of Science and Technology
Tehran-Iran
E-mail:hkorayem@iust.ac.ir

Ali Shahri
Faculty of Electronic Engineering
Iran University of Science and Technology
Tehran-Iran
E-mail:shahri@iust.ac.ir

H.Reza Momeni
Faculty of Technical & Engineering
Tarbiyat Modares University
Tehran-Iran
E-mail:momeni_h@modares.ac.ir

Abstract

Obviously navigation is one of the most complicated issues in mobile robots. Intelligent algorithms are often used for error handling in robot navigation. This Paper deals with the problem of Inertial Measurement Unit (IMU) error handling by using Extended Kalman Filter (EKF) as an Expert Algorithms. Our focus is put on the field of mobile robot navigation in the 2D environments. The main challenge in this issue is to keep track of the position and orientation within a global frame of reference using a variety of sensors providing Dead-Reckoned Odometry, Inertial and Absolute data.

Keywords: *Inertial navigation system, Extended Kalman filter, Error handling.*

1. Introduction

Each mobile object that is free to move in space has six "degrees of freedom" - or ways it can move. There are three linear degrees of freedom (x,y,z) that specify its position and three rotational degrees of freedom (theta (pitch), psi (yaw), and phi (roll)) that specify its attitude. If we know these six variables, we know where it is and which way it is pointed. If we know them over a period of time, then we can also figure out how fast it is moving, and what its acceleration rate is. In fact Navigation System is part of a mobile object to tell it where it is and what it is its' attitude.

From inertial measurements we can determine an estimate for linear accelerations and angular velocities. By integrating these quantities we determine the velocity vector and the body attitude. Position can be calculated by integration of the velocity vector. Inertial navigation is thus based on the dead-reckoning principle[1].

An IMU is a "clump" of six inertial sensors. Three linear accelerometers and three rate gyros make up an IMU. Usually, an IMU also contains a computational unit to do the position calculations based off of the sensors. The operation to combine information from such multi-modal sensors is called sensory fusion. A microcontroller

is often used to interface with sensors and generate control actions based on information gathered from sensors. For reliability and completeness, more than one sensor is generally used[2].

The Kalman Filter (KF) arose out of R.E. Kalman's interest in applying the concept of state vectors to the Wiener filtering problem[3] and it is widely used in aeronautics and engineering for two main purposes: for combining measurements of the same variables but from different sensors, and for combining an inexact forecast of a system's state with an inexact measurement of the state. In fact Kalman filters can be used to derive the best estimation by combining sensory input from different sources[4].

2. Hardware and Software

We performed our experiments by using NAJI2, a three wheels mobile robot, as a test pet and 3DM-GX1^{TM}, an IMU, as a measurement system. The 3DM-GX1^{TM} can output orientation information in three different forms, Euler Angles, Quaternions, or a 3*3 rotation matrix. These are essentially equivalent except that the Euler Angles have a mathematical singularity whenever Pitch is +/-90 degrees, and are therefore unsuitable for use under conditions where such orientations are likely to occur. 3DM-GX1™ combines three angular rate gyros with three orthogonal DC accelerometers, three orthogonal magnetometers, multiplexer, 16 bit A/D converter, and embedded microcontroller, to output its orientation in dynamic and static environments. Operating over the full 360 degrees of angular motion on all three axes, 3DM-GX1™ provides orientation in matrix, quaternion and Euler formats. The digital serial output can also provide temperature compensated calibrated data from all nine orthogonal sensors at update rates of 350 Hz. For more information please refer to www.microstrain.com

A navigation panel, a Graphic User Interface (GUI), was designed by LabView® and the implementations of algorithms were done by MATLAB Version 7.

3. Error handling process:

The process of error handling is divided into two following phases: Error detection and Error correction. In order to test NAJI2 was programmed for cycling in 158 cm diameter to provide a testing motion with constant velocity on the flat floor. A red marker was fixed on the head of the NAJI2 to draw its path.

3.1 Error detection process:

The robot motion was too slow and in a constant velocity we let the acceleration to be zero. First the progress was not successful because it performed for one pride and we couldn't find the error function. So in the second attempt it cycled for six prides and then after calculating the error function the proper result was found and the process of error detection was done. The figure 1, 2 illustrates the passed path on the X, Y axis before error correction. It was not come back to zero.

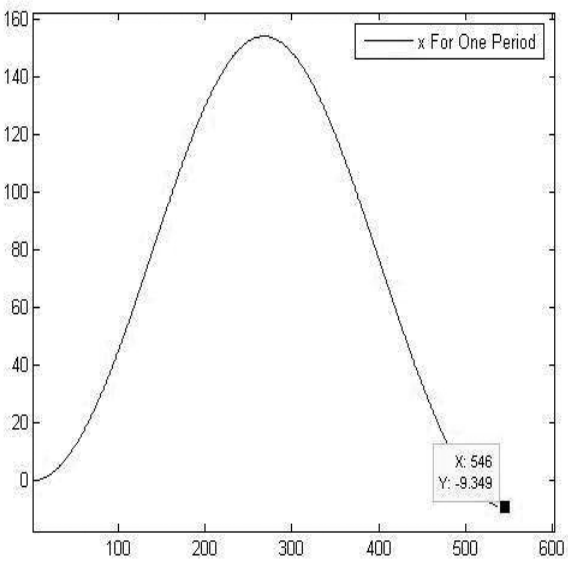

Figure 1. *passed path on the X axis before error correction*

$\Delta x = -9.349$

$n = 546$

$error = \frac{\Delta x}{n} = -0.0171227106$

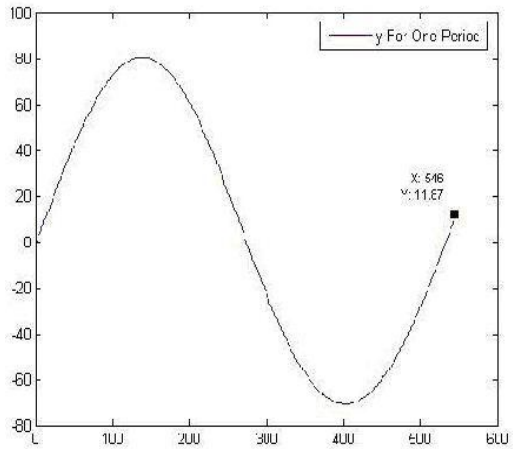

Figure 2. *The passed path on the Y axis before error correction*

$\Delta y = 11.86$

$n = 546$

$error = \frac{\Delta y}{n} = 0.0217216117$

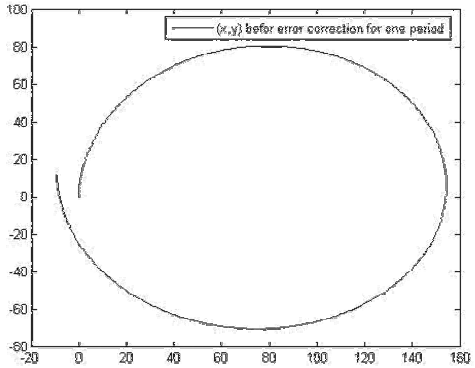

Figure 3. *The passed path before error correction for on period*

3.2 Error correction

The error can be corrected. It was done first for one period and then for six periods. The average of error was calculated and we found the cosine function is down trended by an approximate linear angel .So to correct the deviation we need to calculate this kind of error just by the total number of sampling and the amount of deviation for X,Y separately.

$$\text{The amount of deviation} = \Delta x = -57.29$$
$$\text{The total number of sampling} = n = 3245$$
$$error = \frac{\Delta x}{n} = -0.0176548536$$
$$\text{The amount of deviation} = \Delta y = 80.22$$
$$\text{The total number of sampling} = n = 3245$$
$$error = \frac{\Delta y}{n} = 0.0247211094$$

The error shows us the amount of error in each sample and we must add the absolute value of error to the X and Y separately. The following figures illustrate these error corrections.

Figure 6 provide the motion profile and it illustrates the comparison of motion profile before and after error correction process.

4. Vehicle Models and Odometry

As we allowed the vehicle to move on 2D surface (a floor) and point in arbitrary directions. We can parameterize the vehicle pose x_v(the joint of position and orientation) as:

$$X_V = \begin{bmatrix} x_V \\ y_V \\ \theta_V \end{bmatrix} \qquad (1)$$

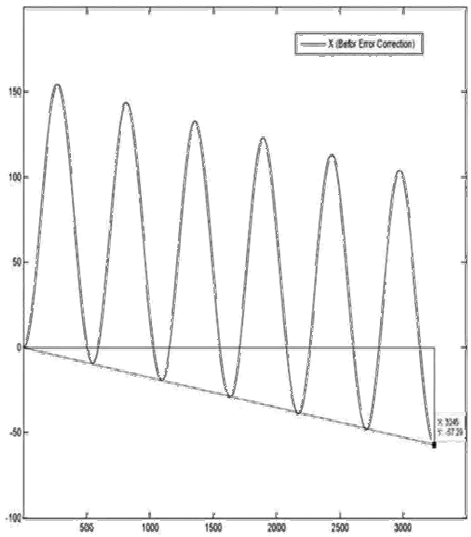

Figure 4. *Calculating the error average on the X axis*

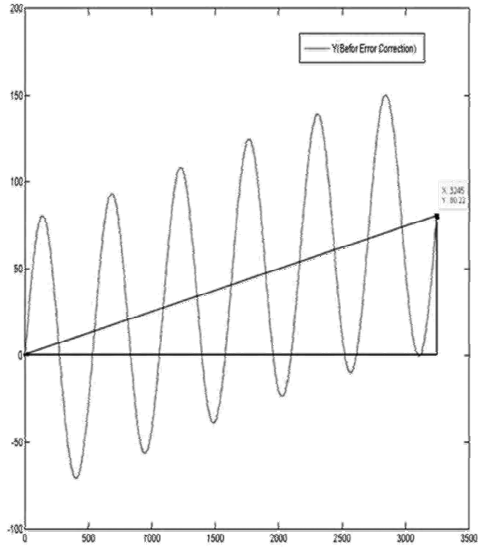

Figure 5. *Calculating the error average on the Y axis*

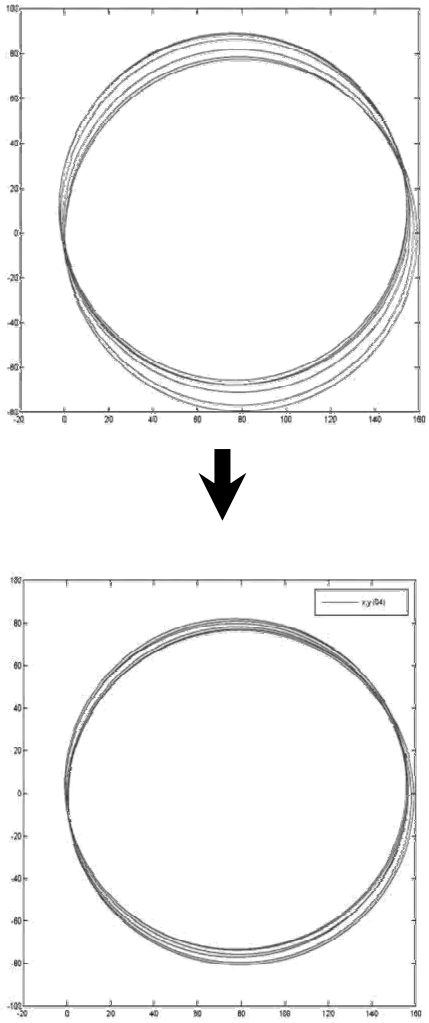

Figure 6. *Comparing the motion profile before and after error correction*

Figure 7 is a diagram of NAJI2, a non-holonomic (local degrees of freedom less than global degree of freedom[5] vehicle with “Ackerman" steering. The angle of the steering wheels is given by ϕ and the instantaneous forward velocity (sometimes called throttle) is V. So we can say

$$\begin{cases} \dot{x}_V = V\cos(\theta_V) \\ \dot{y}_V = V\sin(\theta_V) \end{cases}$$

Using the instantaneous center of rotation we can calculate the rate of change of orientation as a function of steer angle:

$$\frac{L}{a} = \tan(\phi)$$

$$a = \frac{L}{\tan(\phi)} \tag{2}$$

$$a\dot{\theta}_V = V \tag{3}$$

$$\dot{\theta}_V = \frac{V}{L}\tan(\phi) \tag{4}$$

We can now discretise this model by inspection:

$$X_V(k+1) = f(X_V(k), u(k)) \tag{5}$$

$$u(k) = \begin{bmatrix} V(k) \\ \phi(k) \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} x_V(k+1) \\ y_V(k+1) \\ \theta_V(k+1) \end{bmatrix} = \begin{bmatrix} x_V(k) + \delta TV(k)\cos(\theta_V(k)) \\ x_V(k) + \delta TV(k)\sin(\theta_V(k)) \\ \theta_V(k) + \frac{\delta TV(k)\tan(\phi(k))}{L} \end{bmatrix} \tag{7}$$

Note that we have started to lump the throttle and steer into a control-this makes sense if you think about the controlling actions of a human driver. Last Equation is model for a perfect, noiseless vehicle. Clearly this a little unrealistic-we need to model uncertainty[6]. One popular way to do this is to insert terms into the control signal u such that

$$u(k) = u_n(k) + \nu(k) \tag{8}$$

where $u_n(k)$ is a nominal (intended) control signal and $\nu(k)$ is a zero mean Gaussian distributed noise vector:

$$\nu(k) \sim \mathrm{N}(0, \begin{bmatrix} \sigma_V^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}) \tag{9}$$

$$u(k) \sim \mathrm{N}(u_n(k), \begin{bmatrix} \sigma_V^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}) \tag{10}$$

This completes a simple probabilistic model of a vehicle. We shall now see how propagation of this model affects uncertainty in vehicle pose over time.

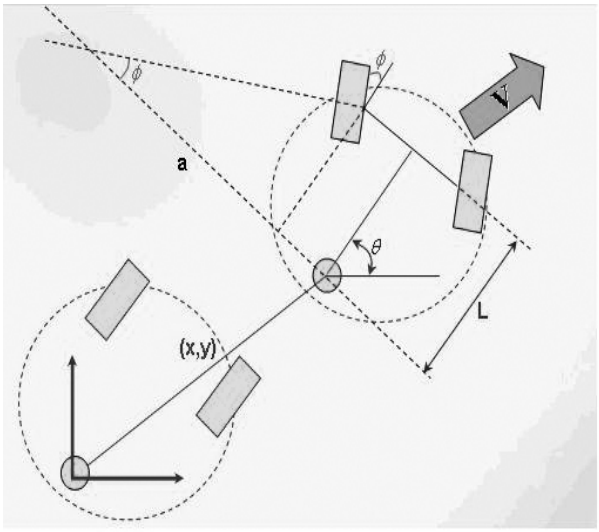

Figure 7. *A non-holonomic vehicle with Ackerman steering*

5. Evolution of Uncertainty

We will examine how an initial uncertainty in vehicle pose increases over time as the vehicle moves when only the control signal u is available. The model derived in the previous section is non-linear and so we will have to use the non-linear form of the prediction step.

Assume at time k we have been given a previous best estimate of the vehicle pose $\hat{x}_v(k-1|k-1)$ and an associated covariance $P_v(k-1|k-1)$.

$$X_v(k|k-1) = f(\hat{X}(k-1|k-1), u(k), k) \tag{11}$$

$$P_V(k|k-1) = \nabla F_X P_V(k-1|k-1) \nabla F_x^T + \nabla F_V Q \nabla F_V^T \tag{12}$$

In this case

$$Q = \begin{bmatrix} \sigma_V^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \tag{13}$$

We need to evaluate the Jacobians with respect to state and control noise at $\hat{x}_v(k-1|k-1)$. We do this by differentiating each row of f by each state and each control respectively:

$$\nabla F_X = \begin{bmatrix} 1 & 0 & -\delta T \sin(\theta_v) \\ 0 & 1 & \delta T \cos(\theta_v) \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

$$\nabla F_u = \begin{bmatrix} \delta T \cos(\theta_v) & 0 \\ \delta T \sin(\theta_v) & 0 \\ \dfrac{\delta T \tan(\phi)}{L} & \dfrac{\delta T \sec^2(\phi)}{L} \end{bmatrix} \tag{15}$$

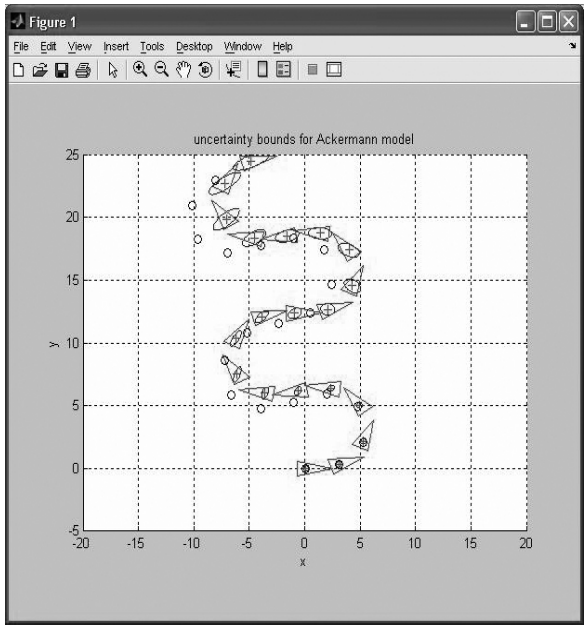

Figure 8. *Uncertainty bounds for Ackerman model*

The Figure 8 shows the results of iterating equations 11 and 12. The circles are the true location of the vehicle whereas the crosses mark the dead-reckoned locations. The orientation of the vehicle is made clear by the orientation of the triangles. Note the divergence between true and dead-reckoned locations. This is typical of all dead reckoning methods. The only thing that can be changed is the rate of divergence. Things are pretty much as we might expect. The uncertainty injected into the system via the noisy control makes the estimated covariance of the vehicles grow without bound[7].

There is an important point to make here that we must understand. In actual real life the real robot is integrating the noisy control signal. The true trajectory will therefore always drift away from the trajectory estimated by the algorithms running inside the robot. This is exactly the same as closing our eyes and trying to walk across University Parks. Our inner ears and legs give you u which we pass through our own kinematics model of our body in our head. Of course, one would expect a gross accumulation of error as the time spent walking “open loop" increases. The point is that all measurements such as velocity and rate of turn are measured in the vehicle frame and must be integrated, along with the noise on the measurements. This always leads to what is called “dead reckoning drift". Figure 9 shows the effect of integrating odometry on NAJI2.The main cause of this divergence on land vehicles is wheel slip. Typically robot wheels are fitted with encoders that measure the rotation of each wheel. Position is then an integral-function of these “wheel counts". The problem is a wheel or radius r may have turned through θ but due to slip/skid the distance traveled over the ground is only $(1-\eta)r\theta$ where η is an unobservable slip parameter.

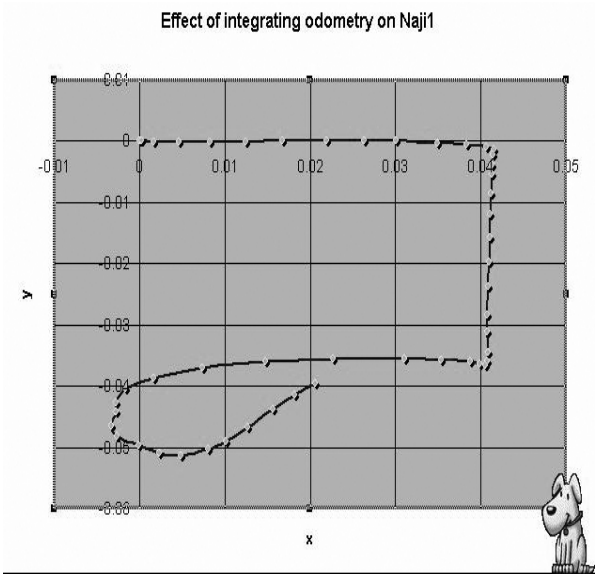

Figure 9. *The effect of integration Odometry*

The Dead Reckoned position from NAJI2 just confused us! The start and end locations are actually the same place! See how we could roll the trajectory back over itself. This is typical of dead reckoned trajectories - small angular errors integrate to give massive long term errors

6. Using Dead-Reckoned Odometry Measurements

The model in the pervious section used velocity and steer angles as control input into the model. It is common to find that this low level knowledge is not easy to obtain or that the relationship between control, prior and prediction is not readily discernable. The architecture in figure 10 is a typical example.

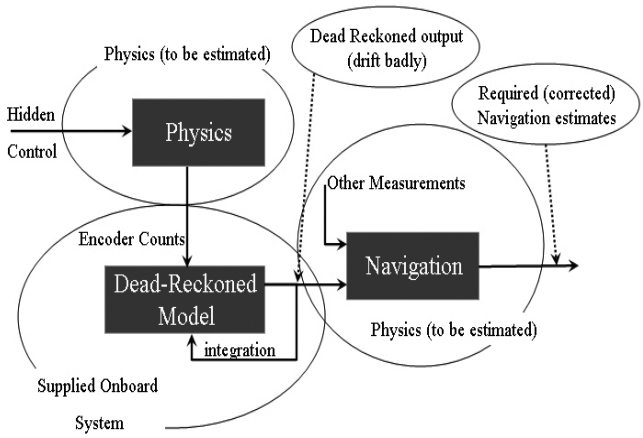

Figure 10. *Use Dead-Reckoned Odometry Measurements*

Sometimes a navigation system will be given a dead reckoned position as input without recourse to the control signals that were involved. Nevertheless the dead-

reckoned position can be converted into a control input (a stream of small motions) for use in the core navigation system.

It would clearly be a bad plan to simply use a dead-reckoned odometry estimate as a direct measurement of state in something like a Kalman Filter. Consider Figure 9 which is the dead reckoned position of NAJI2 moving around some corridors. Clearly by the end of the experiment we cannot reasonably interpret dead-reckoned position as an unbiased measurement of position!

The low level controller on the vehicle reads encoders on the vehicle's wheels and outputs an estimate (with no metric of uncertainty) of its location. We can make a guess at the kind of model it uses. Assume it has two wheels (left and right), radius r mounted either side of its center of mass which in one time interval turn an amount $\delta\theta_l, \delta\theta_r$ - as shown in Figure 10. We align a body-centered co-ordinate frame on the vehicle as shown. We want to express the change of position of the center of the vehicle as a function of $\delta\theta_l, \delta\theta_r$:

$$r\delta\theta_r = (c - L/2)\alpha \tag{16}$$

$$r\delta\theta_l = (c - L/2)\alpha \tag{17}$$

$$\Rightarrow c = \frac{L}{2}\frac{\delta\theta_l + \delta\theta_r}{\delta\theta_l - \delta\theta_r} \tag{18}$$

$$\Rightarrow \alpha = \frac{2r}{L}(\delta\theta_l - \delta\theta_r) \tag{19}$$

Immediately then we have

$$\begin{bmatrix} dx \\ dy \\ d\theta \end{bmatrix} = \begin{bmatrix} (1-\cos\alpha)c \\ c\sin\alpha \\ -\alpha \end{bmatrix} \tag{20}$$

Which for small α becomes:

$$\begin{bmatrix} dx \\ dy \\ d\theta \end{bmatrix} = \begin{bmatrix} 0 \\ r(\delta\theta_l + \delta\theta_r)/2 \\ \dfrac{-2r(\delta\theta_l - \delta\theta_r)}{L} \end{bmatrix} \tag{21}$$

The dead-reckoning system in the vehicle simply compounds these small changes in position and orientation to obtain a global position estimate. Starting from an initial nominal frame at each iteration of its sensing loop it deduces a small change in position and orientation, and then “adds" this to its last dead-reckoned position. Of course the “addition" is slightly more complex than simple adding (otherwise the x coordinate

would always be zero!). What actually happens is that the vehicle composes successive co-ordinate transformation. This is an important concept and will be discussed in the next section.

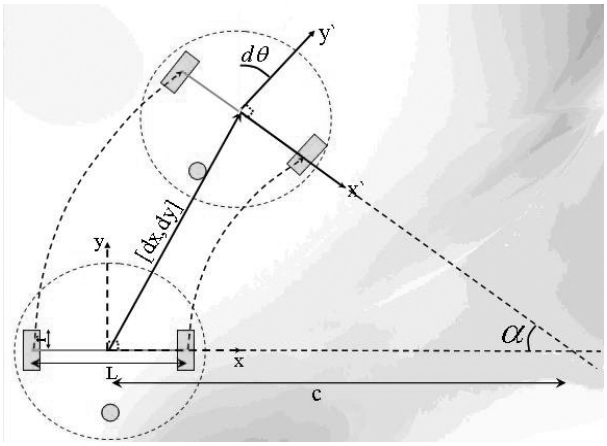

Figure 11. *Geometric Construction for a two wheel drive vehicle*

7. Composition of Transformations

Figure 7 shows three relationships between three coordinate frames. We can express any coordinate j frame with respect to another frame i as a three-vector $x_{i,j}=[xy\theta]$. Here x and y are translations in frame i to a point p and θ is anti-clockwise rotation around p. We define two operators $\oplus$ and Θ to allow us to compose (chain together) multiple transformations:

$$x_{i,k} = x_{i,j} \oplus x_{j,k} \tag{22}$$

$$x_{j,i} = \Theta x_{i,j} \tag{23}$$

With reference to figure 12 we see that $x_{1,3} = x_{1,2} \oplus x_{2,3}$ .But what exactly are these operators? Well, they are just a short hand for a function of one or two transformations:

$$x_1 \oplus x_2 = \begin{bmatrix} x_1 + x_2\cos\theta_1 - y_2\sin\theta_1 \\ y_1 + x_2\sin\theta_1 + y_2\cos\theta_1 \end{bmatrix} \tag{24}$$

$$\Theta x_1 = \begin{bmatrix} -x_1\cos\theta_1 - y_1\sin\theta_1 \\ x_1\sin\theta_1 - y_1\cos\theta_1 \\ -\theta_1 \end{bmatrix} \tag{25}$$

These questions allow us to express something (perhaps a point or vehicle) described in one frame, in another alternative frame. We can use this notation to explain the compounding of odometry measurements. Figure 12 shows a vehicle with a prior pose $x_0(k) = x_0(k-1) \oplus u(k)$. The processing of wheel rotations between successive readings has indicated a vehicle-relative transformation (i.e. n the frame of the vehicle) u. The task of combining this new motion $u(k)$ with the old dead-reckoned estimate x_0 to arrive at a new dead-reckoned pose x_0 is trivial. It is simply:

$$x_0(k) = x_0(k-1) \oplus u(k) \qquad (26)$$

We have now explained a way in which measurements of wheel rotations can be used to estimate dead-reckoned pose. However we set out to figure out a way in which a dead-reckoned pose could be used to form a control input or measurement into a navigation system. In other words we are given from the low-level vehicle software a sequence $x_0(1), x_0(2)...x_0(k)$ etc and we want to figure out $u(k)$. This is now simple and we can invert equation 6.27 to get

$$u(k) = \Theta x_0(k-1) \oplus x_0(k) \qquad (27)$$

Just by looking at the Figure 12 we can see that the transformation $u(k)$ is equivalent to going back along $x_0(k-1)$ and forward along $x_0(k)$. This gives us a small control vector $u(k)$ derived from two successive dead-reckoned poses that is suitable for use in another hopefully less error prone navigation algorithm. Effectively equation 27 subtracts out the common dead-reckoned gross error - locally odometry is good - globally it is poor.

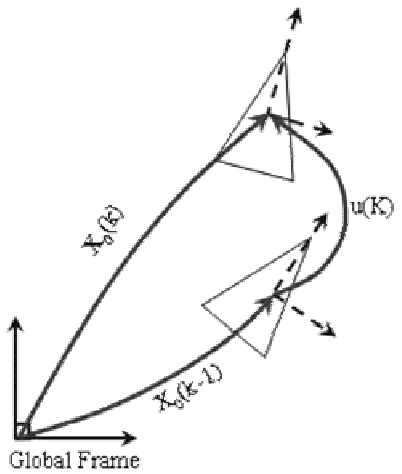

Figure 12. *Using transformation compositions to compound a local odometry measurement with a prior dead-reckoned estimate to deduce a new dead-reckoned estimate*

We are now in a position to write down a plant model for a vehicle using a dead reckoned position as a control input:

$$x_V(k+1) = f(x_V(k), u(k)) \tag{28}$$

$$= x_V(k) \oplus (\underbrace{\Theta x_0(k-1) \oplus x_0(k)}_{dr-control}) \tag{29}$$

$$= x_V(k) \oplus u_0(k) \tag{30}$$

It is reasonable to ask "how dose an initial uncertainty in vehicle pose P_v propagates over time. We know that one way to address this question is to propagate the second order statistics (covariance) of a pdf for x_v through f using following equation:

To do this we need to figure out the Jacobians of equation 30 with respect to xv and u. This is one area where the compositional representation we have adopted simplifies matters. We can define and calculate the following Jacobians the equation was explained before:

$$P(k|k-1) = \nabla F_x P(k-1|k-1) \nabla F_x^T + \nabla G_v Q \nabla G_V^T \tag{31}$$

To do this we need to figure out the Jacobians of equation 30 with respect to x_v and u. This is one area where the compositional representation we have adopted simplifies matters. We can define and calculate the following Jacobians:

$$J_1(x_1, x_2) = \frac{\partial(x_1 \oplus x_2)}{\partial x1} \tag{32}$$

$$= \begin{bmatrix} 1 & 0 & -x_2 \sin\theta_1 - y_2 \cos\theta_1 \\ 0 & 1 & -x_2 \cos\theta_1 - y_2 \sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{33}$$

$$J_2(x_1, x_2) = \frac{\partial(x_1 \oplus x_2)}{\partial x2} \tag{34}$$

$$= \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{35}$$

This allows us to write (substituting into equation 4.31):

$$P(k|k-1) = J_1(x_v, u_0) P_v(k-1|k-1) J_1(x_v, u_0)^T + J_2(x_v, u_0) U_0 J_2(x_v, u_0)^T \tag{36}$$

Were the matrix U_0 describes the strength of noise in the small shifts in pose represented by u_0 derived from two sequential dead-reckoned poses. A simple form of this matrix would be purely diagonal:

$$U_0 = \begin{bmatrix} \sigma_{ox}^2 & 0 & 0 \\ 0 & \sigma_{oy}^2 & 0 \\ 0 & 0 & \sigma_{o\theta}^2 \end{bmatrix} \quad (37)$$

where the diagonals are variances in odometry noise. For example if the odometry loop ran at 20Hz and the vehicle is moving at 1m/s the magnitude of translation in u would be 5cm. If we say slip accounts for perhaps one percent of distance traveled we might "try" a value of $\sigma_{ox}^2 = \sigma_{oy}^2 = (0.05/100)^2$. Allowing a maximum rotation of w perhaps a good starting guess for $\sigma_{o\theta}^2$ would be $(W/100)^2$. These numbers will give sensible answers while the vehicle is moving but not when it is stopped. Even when $u_0 = 0$ the covariance P_v will continue to inflate. This motivates the use of a time varying U_0 which is a function of $u_0(k)$.

8. Conclusion

In the experimental implementation for Kalman Filter we found that we can't depend on the estimated values at the primary states and we were looking for a solution to improve the convergences speed of the algorithm. We found that it can be solved just by changing the transmission matrix from the reference coordinations to global frame. Because by changing the transmission matrix the Kinematics Model can be changed and as the Kalman Filter is related to the Kinematics Model the filter have different outputs. Filter Performance can be improved just by calibrating the IMU and it is argued that computing the INS attitude using quaternion has more advantages than using Euler angles and the direction cosine matrix.

Kalman Filter can be a good linear filter but when we deal with the non-linear problems the Extended Kalman Filter can be a good solution but not all.

In our experimental implementation of INS algorithms, the performance of the filter was influenced by the choice of the process noise attributes. The noise strengths were chosen by engineering judgment and experience and this is a limitation. A method which is able to identify the strengths of process noise from the collected raw data is need.

Figure 13. *NAJI2 was programmed to cycle around the chair.*

References:

[1] Jorge Lobo,Paulo Lucas,Jorge dias,A.Traca de Almeida, ***"Inertial Navigation System for Mobile Land vehicels"***,ISR Instituto of Robotica, University of Coimbra, Portugal

[2] Krishna m. Neaupane, Mitsutaka Sugimoto, ***"An inverse Boundary value problem using the Extended Kalman filter"***, Science Asia,Japan,No.29,pp 121-126,2003.

[3] Kalman, R.E.(1960). "Stochastic Proccesses and Filtering Theory", Journal of Basic Engineering, Series D 82, 34-45

[4] Greg Welch, Gary Bishop, ***"An Introduction to the Kalman Filter"***, Department of Computer Science, University of North Carolina at Chapel Hill

[5] Paul Newman, ***"C4B- Mobile Robotics"***, An Introduction to Estimation and its application to Mobile Robotics, Version 2.00, October 2005, 10-12

[6] Rachel Kleinbauer, ***"Kalman Filtering Implementation with MATLAB®"***, Helsinki University of Technology, November 2004

[7] Grewal M.S., Henderson V. D., Miyasako R. S., ***"Application of Kalman Filtering"***, 3rd Edition, New York, Jon Wiley & Sons, 1997.